疎な多項式の補間のためのモジュラー算法(数式処理と数学研究への応用)

AUTHOR(S):

村尾, 裕一

CITATION:

村尾, 裕一. 疎な多項式の補間のためのモジュラー算法(数式処理と数学研究への応用). 数理解析研究所講究録 1992, 811: 49-59

ISSUE DATE:

1992-10

URL:

http://hdl.handle.net/2433/83031

数理解析研究所講究録
第811巻 1992年 49-59

疎な多項式の補間のためのモジュラー算法

東京大学大型計算機センター 村尾 裕一 (Hirokazu MURAO) [1]
(*E-mail* : murao@cc.u-tokyo.ac.jp)

1. はじめに

疎な多項式を補間により求めるための，実用的な算法が開発されたのは，ここ数年のことである．本稿では，その算法を中国剰余定理と組み合わせ，モジュラー算法へと拡張する方法を示す．同時に，目的とする多項式の，その多項式の決定のために必要な値の個数 (評価の回数) を減らす改良策についても言及する．

1.1. 多項式補間とは?

数式処理における多項式の補間とは，ある多項式を求めることを目的とする計算において，その未知の多項式の値を充分に多くの点において求め，それらの値から目的の多項式を決定する手法である．これにより，数式処理でしばしば問題となる，中間式の爆発的膨張を避けることが可能である．本稿では目的とする多項式は$\boldsymbol{Z}$上の多項式と限定する．我々の問題を具体的に述べると次のとおりである．

多項式補間の問題

$\boldsymbol{Z}$上の多項式 $P(x_1,x_2,\ldots,x_n)$ が，任意の点における値を評価しうるブラックボックスとして与えられた時，$P(x_1,x_2,\ldots,x_n)$ の次数に対する適当な上限 (全次数 $\leq d$) の元に，その多項式を決定せよ．但し，付随的に，各変数 x_i に対する次数の上限値 d_i ($d_i \geq \deg_{x_i}(P)$, $i=1,2,\ldots,n$) や項数の上限値 τ を設けてもよい．

この問題は，今更述べるまでもなく，古くから様々な解法が知られているが，その代表的算法と特徴を列挙すると，以下のとおりである．

- 解法としては，古典的には Newton や Lagrange による補間式が良く知られており，次の特徴が挙げられる．
  - 算法自体は，次数の上限の元で考え得る全ての単項式の，P 中の数係数を求めるに過ぎない．
  - よって，多変数の場合，$\prod_{i=1}^{n}(d_i+1)$ 個という極めて多くの P の値が必要となる．
  - そのために，P がよほど密であると予め判っていない限り，評価の回数が多過ぎて実用上使いものにならない．(P が一変数の場合は密であると仮定して構わない．なぜなら，疎であるならば，適当な冪乗を独立な変数で置き換え，多変数で疎な多項式として扱えば良い．)
- 行列式の展開を初めとした，式の膨張が著しい等途中の計算が複雑な計算において，多項式を扱う計算を数値に対する同等の計算に置き換えることができ，途中の計算の複雑さが軽減される．
- 様々な応用が考えられる．例えば，基本的な計算では，正規多項式表現への簡約化，行列式，逆行列等が挙げられるし，最近の研究 [KT90] によれば GCD や因数分解にも適用可能である．

[1]本研究は，一部，文部省科学技術研究費奨励研究 (A)「スーパーコンピュータと数式処理算法」(課題番号 03780023) の補助を受けて行われました．

1.2. 動機と目的

近年，上記の問題で，多項式が疎である場合について，必要とされる評価の回数を大幅に減らして，充分実用性を持たせた効率の良い算法が開発された(次節参照)．一方，筆者は，スーパーコンピュータを数式処理に応用する研究において，多項式補間を用いた算法が途中の計算を全て数値処理に置き換えることに着目し，中国剰余算法と組み合わせることにより，高効率で有効なベクトル処理が適用可能となり，スーパーコンの豊富な計算機資源を活用できることを示した[Mur91c, Mur91b]．これまでの研究では，一変数の行列式の場合のみを実証的に示したが(全く同じ手法が，既に30年前に，高橋と石橋により発表されていたことが[TI60]後に判明した)，次の段階として，その計算手法を多変数の場合へと拡張するに際し，上記算法をこの応用に適した形へと拡張することが必要となった．本稿の目的は，実際にその拡張を行い，その算法を示すことである．即ち，

我々の問題

前記の問題に，次の点を付け加える．P 中の数係数の絶対値の上限値 C が与えられるとする．算法としては，その積が $> 2C$ となるような複数の法 q に対し，$P \bmod q$ を求めた後，中国剰余算法を適用して $\boldsymbol{Z}$ 上の P を求める，というモジュラー算法を開発せよ．

具体的には，このモジュラー算法を開発する上で，以下の点について論ずる．

- モジュラー写像を用いた場合何が問題となるか，換言すれば，Kaltofen らの拡張[KLW90]が何故に確率的算法になってしまうのか，を明確にし，その対処法を示す．
- 同時に，行列式の展開等の実際の問題では，評価に最も時間を要することが多いことを考慮し，評価回数をさらに減らす方法も示す．
- 以上をまとめ，疎な多項式補間のためのモジュラー算法を新たに開発する．

2. 疎な多項式に対する補間アルゴリズムの最新の動向

我々の問題に入る前に，ここ数年で開発された多項式補間のための算法を以下に列挙し，その特徴と要点をまとめておこう．

- Ben-Or と Tiwari の算法 (1988年 STOC) [BT88] — 詳細は次節．
  - 求める多項式 P 中の項数に対し，上限値 τ のみを設定する．
  - 多項式の評価には，Grigoriev と Karpinski [GK87] による値を用いる．これにより次のことが可能となる．
  - その値の列から正則な Toeplitz 行列を構成する．その次数が t を与え，また，その方程式の解から作られる一変数多項式の線形因子から，P 中の全ての単項式が決定される．
  - 各数係数は，単項式と多項式両方の値から構成される転置 Vandermonde 方程式を解くことにより得られる．
- Kaltofen と Lakshman (1988年 ISSAC '88) [KL88]
  前項の算法中の各ステップに，効率の良い既知の算法を適用した．
  - Toeplitz 方程式の解法に Brent らによる算法[BGY80]を用いる．

- 転置 Vandermonde 方程式の解法に，古くから知られた算法 [Zip90] を適用することを提案し，計算量を改善した．

- Kaltofen, Lakshman と Wiley (1990 年 ISSAC '90) [KLW90]
  - 上記算法では，特に Toeplitz 行列の扱いにおいて，数係数の膨張が，実用性を損なうほどに，著しいことを報告した．
  - この問題を回避するために，$\bmod p^k$ によるモジュラー算法を開発した．
    * 但し，法 p によっては失敗する．
    * その確認は，求めた多項式の値と評価の結果の値との比較等による．
    * 失敗する確率を充分に小さくするための，法の選択法を示した．
    * さらに，$\boldsymbol{Q}$ 上の多項式を求めるために，Wang らの方法 [WGD82] を適用した．

- 村尾 (1991 Sep. ...) [Mur91a] — 本稿．

但し，ここで取り上げた算法は，Ben-Or と Tiwari の算法に基づいたもののみで，[Zip79] や [Zip90] のように，疎であることを考慮して，古典的な算法に確率的手法を持ち込んだり，その対処を行ったものは除外した．

3. Ben-Or と Tiwari の算法

本節では，我々の拡張の基礎となる，Ben-Or と Tiwari による算法を示す．先ず，そのための記法を若干用意する．但し，P 及び τ は冒頭のとおりとする．加えて，P 中の本当の項数を $t\ (\leq \tau)$ とし，求める多項式表現を $P(x_1, x_2, \ldots, x_n) = \sum_{k=1}^{t} c_k x_1^{e_{k,1}} x_2^{e_{k,2}} \cdots x_n^{e_{k,n}}$ とする (すべて未知)．小さい方から i 番目の素数 $(2, 3, 5, \ldots)$ を p_i で表し，$P(x_1, x_2, \ldots, x_n)$ を $({p_1}^j, {p_2}^j, \ldots, {p_n}^j)$ で評価した値を

$$a_j \Leftarrow P({p_1}^j, {p_2}^j, \ldots, {p_n}^j) = \sum_{k=1}^{t} c_k {b_k}^j,$$

とおく．但し，b_k は各単項式を評価した値 $b_k = p_1^{e_{k,1}} p_2^{e_{k,2}} \cdots p_n^{e_{k,n}}$ である．さらに，a_j を要素とする $u \times u$ の Toeplitz 行列 $\boldsymbol{A}_u$ と列ベクトル $\vec{a}_{u,v}$ を定義する：

$$\boldsymbol{A}_u = \begin{pmatrix} a_0 & a_1 & \ldots & a_{u-1} \\ a_1 & a_2 & \ldots & a_u \\ \vdots & \vdots & \ddots & \vdots \\ a_{u-1} & a_u & \ldots & a_{2u-2} \end{pmatrix}, \quad \vec{a}_{u,v} = \begin{pmatrix} a_u \\ a_{u+1} \\ \vdots \\ a_{u+v-1} \end{pmatrix}.$$

図 1に具体的な算法を示す．以下に，そのアイディアの詳細と改良点を列挙する．

- [GK87] で示された評価点の選択法 (各変数に対して異なる素数の冪) を用いることに依り，b_k の値は互いに全て異なり，また b_k の素因数分解により指数 $(e_{k,1}, e_{k,2}, \ldots, e_{k,n})$ が定められるようになる．

- Toeplitz 行列 $\boldsymbol{A}_u$ は，$u = t$ ならば正則，$u > t$ ならば非正則である．何故なら，

$$\boldsymbol{V}_t = \begin{pmatrix} 1 & 1 & \ldots & 1 \\ b_1 & b_2 & \ldots & b_t \\ \vdots & \vdots & \ddots & \vdots \\ {b_1}^{t-1} & {b_2}^{t-1} & \ldots & {b_t}^{t-1} \end{pmatrix}$$

Ben-Or と Tiwari の算法

入力 : $P \in \boldsymbol{Z}[x_1, x_2, \ldots, x_n]$ 及び τ.

出力 : P の多項式表現 $\sum_{k=1}^{t} c_k x_1^{e_{k,1}} x_2^{e_{k,2}} \cdots x_n^{e_{k,n}}$.

ステップ **1** : $a_j, j = 0, 1, \ldots, 2\tau - 1$ を評価し，行列 $\boldsymbol{A}_\tau$ の階数を求める．その値が t である．

ステップ **2** : 正則な $t \times t$ の Toeplitz 方程式 $\boldsymbol{A}_t \vec{\lambda}_t = -\vec{a}_{t,t}$ を解き，$\vec{\lambda}_t = (\lambda_1, \lambda_2, \ldots, \lambda t)^T$ を求める．

ステップ **3** : 一変数多項式 $\Lambda(z) = 1 + \sum_{i=1}^{t} \lambda_i z^i = \prod_{k=1}^{t} (1 - b_k z)$ を構成し，因数分解の結果 b_k を得る．各 b_k を素因数分解 ($p_i, i = 1, 2, \ldots, n$ による除算でよい) し，P 中の各単項式の指数 $(e_{k,1}, e_{k,2}, \ldots, e_{k,n})$ を求める $(k = 1, 2, \ldots, t)$.

ステップ **4** : 最後に，下記の転置 Vandermonde 方程式を解き，数係数 c_i $(1 \leq i \leq t)$ を定める．

$$\begin{pmatrix} 1 & 1 & \ldots & 1 \\ b_1 & b_2 & \ldots & b_t \\ \vdots & \vdots & \ddots & \vdots \\ {b_1}^{t-1} & {b_2}^{t-1} & \ldots & {b_t}^{t-1} \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_t \end{pmatrix} = \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_{t-1} \end{pmatrix}.$$

図 1: Ben-Or と Tiwari の算法

と置けば，

$$\boldsymbol{A}_t = \begin{pmatrix} a_0 & a_1 & \ldots & a_{t-1} \\ a_1 & a_2 & \ldots & a_t \\ \vdots & \vdots & \ddots & \vdots \\ a_{t-1} & a_t & \ldots & a_{2t-2} \end{pmatrix} = \boldsymbol{V}_t \begin{pmatrix} c_1 & & & 0 \\ & c_2 & & \\ & & \ddots & \\ 0 & & & c_t \end{pmatrix} \boldsymbol{V}_t^T,$$

ゆえ，

$$\det(\boldsymbol{A}_t) = \prod_{i=1}^{t} c_i \prod_{j<i} (b_i - b_j)^2. \tag{1}$$

よって $\det(\boldsymbol{A}_t) \neq 0$, 即ち $\boldsymbol{A}_t$ は正則である．非正則であることは，全ての $j \geq 0$ に対し

$$a_{j+t} + \sum_{i=1}^{t} \lambda_i a_{j+t-i} = \sum_{k=1}^{t} c_k {b_k}^{j+t} \Lambda({b_k}^{-1}) = 0,$$

となることから示される．また，この線形関係 $(j = 0, 1, \ldots, t-1)$ が $\vec{\lambda}_t$ に対する Toeplitz 方程式を導く．Toeplitz 方程式の解法については，様々な研究成果がある [Bla85, BGY80] .

- 符合理論では，上の線形関係を満たす多項式 $\Lambda(z)$ を，数列 $a_0, a_1, \ldots, a_{2t-1}$ に対する connection polynomial と呼ぶ [Mas69, Bla85] :

$$\Lambda(z) = \prod_{k=1}^{t} (1 - b_k z) = 1 + \sum_{i=1}^{t} \lambda_i z^i. \tag{2}$$

この多項式を導くには，Berlekamp-Massey の算法 [Mas69, Bla85] を用いれば良いことが知られており [BT88]，実際 [KLW90] で用いられた．その算法を用いれば，t と $\Lambda(z)$ が同時に求められるので，Toeplitz 行列を扱う必要はなくなる．

- 転置 Vandermonde 方程式を解くための効率の良い算法が知られており [Zip90, KL88]，その算法は，Vandermonde 行列が正則である限り，モジュラー算法へと直接に拡張可能である．

4. Berlekamp-Massey の算法

Berlekamp-Massey の算法は，本来，符合理論において，最短の linear feedback shift register を設計するための算法である．特別な数列 $a_0, a_1, \ldots, a_{2t-1}$ に対して行われる我々の応用では，最低次の connection polynomial が求められる．図2に算法を示す．

Berlekamp-Massey の算法

入力 : 数列 $s_0, s_1, \ldots, s_{2\tau-1} \in \boldsymbol{K}$，但し，$\boldsymbol{K}$ は任意の体．

出力 : 数列に対する最低次の connection polynomial $\zeta(z) \in \boldsymbol{K}[z]$.

[初期化] $\zeta(z) \Leftarrow 1$; $L \Leftarrow 0$; $\phi(z) \Leftarrow 1$;

[connection polynomial 等を更新するループ]

for $j := 0$ **to** $2\tau - 1$ **do**

begin

$\delta \Leftarrow s_j + \sum_{k=1}^{L} s_{j-k}\zeta_k(z)$; % discrepancy;

if $\delta = 0$ **then** $\phi(z) \Leftarrow z\phi(z)$

else if $2L > j$ **then** $\begin{pmatrix} \zeta(z) \\ \phi(z) \end{pmatrix} \Leftarrow \begin{pmatrix} \zeta(z) - \delta z\phi(z) \\ z\phi(z) \end{pmatrix}$

else $\begin{pmatrix} \zeta(z) \\ \phi(z) \\ L \end{pmatrix} \Leftarrow \begin{pmatrix} \zeta(z) - \delta z\phi(z) \\ \phi(z)/\delta \\ j+1-L \end{pmatrix}$

end;

図 2: Berlekamp-Massey の算法

この算法を我々の問題に適用する場合，以下の点に注意しなければならない．

- $\boldsymbol{K} = \boldsymbol{Z}/(q)$ の時，$\zeta(z)$ が(2)のモジュラー像 $(\Lambda(z) \bmod q)$ に等しいとは限らない．
- さらに，$\boldsymbol{K} = \boldsymbol{Z}/(q^k)$ 上で適用した場合除算に失敗する可能性がある．しかし，その確率を充分に小さくすることは可能である [KLW90].
- 我々の多項式補間への応用では，必要とされる多項式は，最初の $2t$ の繰り返しで求まってしまい，ループの残りは確認をしているに過ぎない．

5. 中国剰余定理との組合せでモジュラー算法へと拡張するには?

5.1. 留意点および問題点

以下に，モジュラー算法へと拡張する上で問題となる点を列挙する．

- $\boldsymbol{Z}/(q)$ では $\boldsymbol{A}_t$ は必ずしも正則とは限らない．正則でなくなるのは，次の条件のいずれか或は両方が成り立つ場合である：
  1. $c_i \equiv 0 \pmod{q}$ なる i が存在する，
  2. $b_i \equiv b_j \pmod{q}$ となる $i, j\ (i \neq j)$ が存在する．

  この非正則な場合，connection polynomial は如何なる多項式か? また，P 中の全ての単項式を特定することが可能であろうか? ここで，上の非正則性をもたらす添字を除いた添字の集合 σ_q を定義しておく．

$$\sigma_q = \{k \mid 1 \leq k \leq t \wedge c_k \not\equiv 0 \pmod{q} \wedge \forall j < k, b_k \not\equiv b_j \pmod{q}\}$$

- 複数の法 q_l に対する $(b_k \bmod q_l)$ の値から，如何にして指数 $(e_{k,1}, e_{k,2}, \ldots, e_{k,n})$ を求めるか．
- 法の選択
  - $\boldsymbol{Z}$ での数係数を求めるには，$2 \times$ (法の積) $\geq C$ は必須である．
  - 法の選択によって上記の非正則性を検知できるだろうか?
- 異なる i, j に対して $b_i \equiv b_j \pmod{q}$ となる i, j が存在する場合，転置 Vandermonde 行列 $\boldsymbol{V}_t$ も正則でなくなり，係数を決定できない．

5.2. モジュラー算法への拡張の方法

上記の問題点に対する解決策を示し，モジュラー算法へと拡張する方法を述べる．

1. $\boldsymbol{Z}/(q)$ 上での connection polynomial.

   数列 $a_0, a_1, \ldots, a_N\ (N \geq 2t-1)$ に対する，$\boldsymbol{Z}/(q)$ 上の最低次の connection polynomial $\Lambda_q(z)$ は unique で，

$$\Lambda_q(z) = \prod_{k \in \sigma_q} (1 - b_k z) \bmod q$$

   である．このことは，$\boldsymbol{Z}/(q)$ 上において数列が $(\sum_{k \in \sigma_q} c_{k,q} b_k \bmod q)$ のように振舞うことを考えれば明らかである．

2. 未知の正の整数 b に対し，b を求めずに，法 $q_l\ (l = 1, 2, \ldots, s)$ についての剰余 $(b \bmod q_l)$ のみから，その素因数分解 $p_1{}^{e_1} p_2{}^{e_2} \cdots p_n{}^{e_n}$ を得るには，剰余 $(b \bmod q_l)$ を変換し b の混合基数表示 [Knu81, 273–275 ページ] を用いれば良い．但し，素因数 $p_1, p_2, \ldots p_n$ は既知とし，$\prod_{l=1}^{s} q_l > b$ とする．

   (a) 混合基数表示 $b = v_1 + v_2\, q_1 + v_3\, q_1 q_2 + \cdots + v_s\, q_1 q_2 \cdots q_{s-1}$ における剰余の組 $(v_1, v_2, \ldots, v_s)$ を，次のようにして求める

$$\begin{aligned} v_1 &= b \bmod q_1 \\ v_j^{(1)} &= b \bmod q_j, \qquad v_j^{(i+1)} = (v_j^{(i)} - v_i) q_i{}^{-1} \bmod q_j, \qquad v_j = v_j^{(j)}. \end{aligned}$$

(b) 各々の $p_i, i = 1, 2, \ldots, n$ に対し，以下を繰り返す．但し，p_i は p と略す．

i. q_k を p で割った商と余りを各々 γ_k, δ_k とする $(q_k \Rightarrow \gamma_k p + \delta_k,\ 1 \leq k \leq s-1)$. $e_i \Leftarrow 0$.

ii. $v_1 + \sum_{j=1}^{s-1} v_{j+1} \prod_{k=1}^{j} \delta_k \bmod p$ が 0 に等しくなければ，次の素数 p_{i+1} へ．

iii. $e_i \Leftarrow e_i + 1$. $(v_1, v_2, \ldots, v_s)$ を，p_i で割った商へと，更新する．

$$\begin{pmatrix} v_s \\ r \end{pmatrix} \Leftarrow \begin{pmatrix} \lfloor v_s/p \rfloor \\ v_s \bmod p \end{pmatrix}.$$

$$\begin{pmatrix} v_j \\ r \end{pmatrix} \Leftarrow \begin{pmatrix} \lfloor (\delta_j r + v_j)/p \rfloor + \gamma_j r \\ \delta_j r + v_j \bmod p \end{pmatrix}, \qquad j = s-1, s-2, \ldots, 1.$$

3. P 中の単項式とその個数 t を定めるには?

$\boldsymbol{Z}/(q)$ 上の Toeplitz 行列や上の $\Lambda_q(z)$ から，t や P 中の全単項式を決定することは，次の理由により，殆んど不可能に近い．

- 単項式の値 b_k の取り得る値の最大値を

$$B = \max_{0 \leq e_j \leq d_j,\ \sum_{j=1}^{n} e_j \leq d} \{ p_1^{e_1} p_2^{e_2} \cdots p_n^{e_n} \}$$

と置く．法の積 $\prod_l q_l$ が

$$\prod_l q_l > C B^{\tau(\tau-1)/2} \geq C B^{t(t-1)/2} > C \prod_{j \neq k} |b_j - b_k|$$

であれば，(1)より，全ての法 q_l に対して $\det(\boldsymbol{A}_t)$ が 0 とはなり得ず，いずれかの法で t が $\deg(\Lambda_{q_l}(z))$ として求まる．しかし，上の法の積に対する条件は，大抵の場合，大き過ぎて非実用的である．

- たとえ t が正確に得られたとしても，異なる q_l の間で $(b_k \bmod q_l)$ の対応が定まらず，その組合せが多大であるのみならず，t 個以上の指数が求まり得る．

前者の問題については，法を適切に選べば，高い確率で t が得られることが解っており，そのため，最終的な答が正しくないと判明した場合でも，法を新たに選び直して算法を繰り返せば，実用上はあまり問題にはならない．[KLW90]．しかし，数値による評価の段階と数式を構築する段階とを明確に分離する我々の実現手法 [Mur91c] では，そのような確率的算法は，その分離を損なうことにつながり，不適切である．しかも，失敗が判明した場合には，新たな法に対して，評価も含めた類似の計算を繰り返すことになり効率が悪い．これらの問題に対する我々の解決策は，上記の二番目の問題点が示唆するように，

「$\Lambda_q(z) = 0$ の根は，P 中の単項式を定めるのではなく，存在し得る単項式を限定する」

と見なすことである．即ち，Ben-Or と Tiwari の算法における「疎な多項式中の単項式を決定する」ための主要なアイディアに固執せずに，疎であることを活かしながら，存在しそうな全ての項に対し数係数を決定するのである (多項式補間の原点に立ち帰ることに他ならない)．実用上は，法を充分に大きくとれば，$\Lambda_q(z) = 0$ の次数は t となり，その根は $b_k \bmod q$ に限定され，疎であることは保存されると予想する．

4. 単項式の限定と法に対する条件

変数の個数が少なく，しかも次数が低ければ，B の値は小さくなり，$q > B$ と選択することが可能な場合もあり得る．その場合には，$c_k \equiv 0 \pmod{q}$ なる k を除いて全ての $(e_{k,1}, e_{k,2}, \ldots, e_{k,n})$ は決定される．さらに，$2 \times$ (法の積) $\geq C$ は必須条件であり，その全ての法が上の条件を満たすならば，前項で掲げた問題は起きず，直接 Ben-Or と Tiwari の方法により指数を決定することが可能である．

また，そのような条件が成り立たなくとも，次数の制限を満たす全ての $(e_1, e_2, \ldots, e_n)$ から，全ての q_l に対し次の条件を満たす組を選び出せば，単項式はかなり限定される：

$$\Lambda_{q_l}({p_1}^{-e_1} {p_2}^{-e_2} \cdots {p_n}^{-e_n}) \equiv 0 \bmod q_l$$

この条件を満たすかの確認は，

(a) [CZ81] の確率的算法により Λ_{q_l} を一次因子に分解し，各因子の数係数を求めておき (算法は確率的でも，全て一次因子まで分解することは既知であるから結果は確定する)，

(b) $(e_1, e_2, \ldots, e_n)$ に対応する $b = {p_1}^{e_1} {p_2}^{e_2} \cdots {p_n}^{e_n}$ の値について，いずれの q_l に対しても，$b \bmod q_l$ が，前項で求めた数係数中に存在するかを調べる

という方法によって，極めて簡単に行われる．この方法は，計算量の観点からは，我々の算法を古典的な手法と同等の $\prod_i (d_i+1)$ にまで劣化させることになるが，上記の値の照合の計算コストは，多項式の評価に比較すれば，遥かに安価である．よって，目的とする多項式の変数の個数をある程度限定すれば，充分に実用性を有するであろう．実際，スーパーコンを用いれば，全ての $b \bmod q$ は，下表のとおり，充分に実用的な計算時間で求められる．

HITAC S-820 (128MB) における全ての $b \bmod q$ を求める計算時間

変数の個数	次数の上限値	CPU 時間 (msec)	
n	d_i	SPU	VPU
2	10, 10	0.355	0.0212
	100, 100	5.896	5.247
	200, 200	41.333	40.8
	400, 400	319.693	323.342
3	10, 10, 10	0.385	0.092
	20, 20, 20	1.312	0.979
	40, 40, 40	14.094	13.927
	100, 100, 100	493.969	517.217
4	10, ..., 10	1.066	0.772
	20, ..., 20	19.954	19.621
	40, ..., 40	506.79	567.65
5	10, ..., 10	8.486	8.192
	20, ..., 20	365.642	410.941
	30, ..., 30	1326.643	4314.426
6	10, ..., 10	849.22	89.763
	20, ..., 20	365.785	430.893
8	10, ..., 10	625.317	1076.387
10	10, ..., 10, 2	19456.989	35495.874

さらに，(法の積) $> B^\tau \geq B^t$ なる条件を満たすだけ多くの法を用いることが可能ならば，$\prod_{k=1}^{t}(b'-b_k)$ は 0 でないが法の積の倍数となるような不要な b' は除外されるので，単項式の限定に寄与する．

5. 係数を決定するには?

それぞれの法 q について，

(a) P 中に存在する可能性のある全ての単行式に対し $b_{k,q}$ を計算する．

(b) $b_{k,q}$ の全ての異なる値 (重複する分は一つのみをとる) から Vandermonde 方程式を構成し，解く．

(c) 値の重複していない $b_{k,q}$ に対応する $c_{k,q}$ は，そのままで求める係数である．

(d) 重複した分については，Vandermonde 方程式の解は，$b_{k,q}$ に等しくなる k に対する $c_{k,q}$ の和を与える．

(e) 今度は存在し得る単項式が定まっているので，P から上で定まった単項式を差し引いた多項式に対し，新たに評価した値から係数を定めれば良い．
(多項式補間の基本)

6. 多項式の評価の回数を減らすには?

少なくとも，$\boldsymbol{A}_t$ が正則であることが判っていれば (整数上，あるいは $q > B$ が成り立つ)，下記のように，Toeplitz 行列の特定の行 / 列を取り出して考えれば，Berlekamp-Massey の算法のループ中で，連続する τ 個の値 $(a_s, a_{s+1}, \ldots, a_{s+\tau-1})$ に対して connection polynomial が不変であれば，最初の $s+1$ 行 / 列に線形関係が存在することに他ならず，その多項式が求めるべき connection polynomial であることが判る．

$$\boldsymbol{A}_\tau = \begin{pmatrix} a_0 & a_1 & \cdots & \boxed{a_s} & \cdots & a_{\tau-1} \\ a_1 & a_2 & \cdots & a_{s+1} & \cdots & a_\tau \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ a_s & a_{s+1} & \cdots & a_{2s} & \cdots & a_{s+\tau-1} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ a_{\tau-1} & a_\tau & \cdots & \boxed{a_{s+\tau-1}} & \cdots & a_{2\tau-2} \end{pmatrix}$$

7. 転置 Vandermonde 方程式の解法

最後に，$\vec{c}_t = (c_1, c_2, \ldots, c_t)^T$ 及び $\vec{a}_{0,t} = (a_1, a_2, \ldots, a_t)^T$ として，既に上で触れた，転置 Vandermonde 方程式 $\boldsymbol{V}_t\vec{c}_t = \vec{a}_{0,t}$ に対する効率の良い解法を示しておこう．

Vandermonde 行列 $\boldsymbol{V}_t$ の転置行列を $\boldsymbol{B} = \boldsymbol{V}_t{}^T$ とし，$(\beta_{0,j}, \beta_{1,j}, \ldots, \beta_{t-1,j})^T$ が $\boldsymbol{B}^{-1}$ の第 j 列ベクトルを表すとする．その要素を係数とする一変数多項式 $B_j(z) = \beta_{0,j} + \beta_{1,j}z + \cdots + \beta_{t-1,j}z^{t-1}$ を定義する．行列の積 $\boldsymbol{B}\boldsymbol{B}^{-1} = \boldsymbol{V}_t{}^T\boldsymbol{B}^{-1}$ は単位行列ゆえ，$B_j(b_i) = \delta_{ij}$．よって，$(t-1)$ 次多項式 $B_j(z)$ は

$$B_j(z) = \prod_{1\leq k\leq t, k\neq j} \frac{z-b_k}{b_j-b_k} = \frac{B(z)}{\alpha_j(z-b_j)}, \quad \text{但し } \alpha_j = \prod_{1\leq k\leq t, k\neq j}(b_j-b_k) = B'(b_j)$$

転置 Vandermonde 方程式の解法

入力 : $\boldsymbol{V}_t$ 及び $\vec{a}_{0,t}$.

出力 : $\vec{c}_t$.

ステップ **1** : 一変数多項式 $B(z)$ を $B(z) \Leftarrow \prod_{k=1}^{t}(z-b_k)$ とおく. この多項式は $\Lambda(z)$ から直接与えられる.

ステップ **2** : さらに, もうひとつの一変数多項式 $D(z)$ を $D(z) = \sum_{k=0}^{t-1} a_k z^{t-k} = a_0 z^t + a_1 z^{t-1} + \cdots + a_{t-1} z$ と定義する.

ステップ **3** : 上の二つの多項式の積 $B(z)D(z)$ 中の z^j $(j = t+1, t+2, \ldots, 2t)$ の係数を求め, 各々を q_j とする. この q_j を係数とする一変数多項式 $Q_t(w)$ を $Q_t(w) = \sum_{j=1}^{t} q_{t+j} w^{j-1}$ と定義する.

ステップ **4** : 解 c_j は, $c_j = Q_t(b_j)/B'(b_j)$ により与えられる.

図 3: 転置 Vandermonde 方程式の解法

となる. すると, 解は $\vec{c}_t = \boldsymbol{V}_t^{-1}\vec{a}_{0,t} = \left(\boldsymbol{B}^{-1}\right)^T \vec{a}_{0,t}$ ゆえ, その各要素 c_j は $\sum_{k=0}^{t-1} \beta_{k,j} a_k$ で与えられる. この値は $B_j(z)D(z)$ 中の z^t に他ならず, $Q_t(b_j)$ により求められる. 図 3が, その算法である.

8. おわりに

本稿では, 記号多項式を求める計算においてベクトル処理を適用するために必要となる, 多項式補間のモジュラー算法を示した. モジュラー算法への拡張では, ベクトル処理という観点からは, 多倍長演算を除去することが最大の目的なのだが, 一方では, 全ての単項式の指数を決定するステップを除けば, 各々の法に対する処理は, 互いに独立であるため, 並列処理が可能となっている. 即ち, 本稿で示したモジュラー算法は, 疎な多項式の補間の問題に対する, 有効な並列処理算法の可能性を示唆している. また, 複数の法を用いる算法では, 数係数の上限値は不当に大きく与えられるのが普通であるから, 通常はかなり多くの法が必要となる. このため, 並列もしくはベクトル処理を適用できない場合には, 多倍長演算を伴うとはいえ, Kaltofen らのモジュラー算法 [KLW90] を用いた方が高速であろう.

本稿の算法は, 単項式の決定において, 古典的な算法と同程度の計算量となることを述べたが, つい最近, この問題も解決されたことだけを [Mur92] 最後に報告しておこう.

参考文献

[BGY80] R. P. Brent, F. G. Gustavson, and D. Y. Y. Yun. Fast solution of Toeplitz systems of equations and computation of Padé approximants. *Journal of Algorithms*, Vol. 1, pp. 259–295, 1980.

[Bla85] R. E. Blahut. *Fast Algorithms for Digital Signal Processing.* Addison-Wesley, 1985.

[BT88] M. Ben-Or and P. Tiwari. A deterministic algorithm for sparse multivariate polynomial interpolation. In *Proceedings of 20th Symposium on the Theory of Computing*, pp. 301–309, 1988.

[CZ81] D. G. Cantor and H. Zassenhaus. A new algorithm for factoring polynomials over finite fields. *Mathematics of Computation*, Vol. 36, pp. 587–592, 1981.

[GK87] D. Yu. Grigoriev and M. Karpinski. The matching problem for bipartite graphs with polynomially bounded permanents is in NC. In *Proc. 28th IEEE Symposium on the Foundations of Computer Science*, pp. 166–172, 1987.

[KL88] E. Kaltofen and Yagati Lakshman. Improved sparse multivariate polynomial interpolation algorithms. In P. Gianni, editor, *Symbolic and Algebraic Computation, ISSAC '88*, number 358 in LNCS, pp. 467–474, Rome, Italy, July 4–8 1988. Springer-Verlag.

[KLW90] E. Kaltofen, Y. N. Lakshman, and J-M. Wiley. Modular rational sparse multivariate polynomial interpolation. In S. Watanabe and M. Nagata, editors, *Proceedings of ISSAC '90*, pp. 135–139, Tokyo, Japan, August 20–24 1990.

[Knu81] D. E. Knuth. *Seminumerical Algorithms*, volume 2 of *The Art of Computer Programming.* Addison-Wesley, 2nd edition, 1981.

[KT90] E. Kaltofen and B. M. Trager. Computing with polynomials given by black boxes for their evaluations: Greatest common divisors, factorization, separation of numerators and denominators. *Journal of Symbolic Computation*, Vol. 9, No. 3, pp. 301–320, 1990.

[Mas69] J. L. Massey. Shift-register synthesis and BCH decoding. *IEEE Transactions on Information Theory*, Vol. IT–15, No. 1, pp. 122–127, 1969.

[Mur91a] H. Murao. Improved modular Ben-Or and Tiwari algorithm for sparse multivariate polynomial interpolation. (manuscript), September 1991.

[Mur91b] H. Murao. Vector processing in symbolic determinant expansion on supercomputer. In *Proc. International Symposium on Supercomputing : ISS '91*, pp. 145–154, Fukuoka, Japan, November 6–8 1991.

[Mur91c] H. Murao. Vectorization of symbolic determinant calculation. *SUPERCOMPUTER*, Vol. VIII, No. 3, pp. 36–48, May 1991.

[Mur92] H. Murao. A deterministic modular algorithm for determination of monomials in black boxes of sparse multivariate polynomials. (in preparation), March 1992.

[TI60] 高橋秀俊, 石橋善弘. 電子計算機による exact な計算の新方法. 情報処理, Vol. 1, No. 2, pp. 78–86, 1960.

[WGD82] P. S. Wang, M. J. T. Guy, and J. H. Davenpont. P-adic reconstruction of rational numbers. *SIGSAM Bulletin*, Vol. 16, No. 2, pp. 2–3, May 1982.

[Zip79] R. E. Zippel. Probabilistic algorithms for sparse polynomials. In E. W. Ng, editor, *Symbolic and Algebraic Computation, Proceedings of EUROSAM '79*, number 72 in LNCS, pp. 216–226, Marseille, France, April 5–7 1979. Springer-Verlag.

[Zip90] R. Zippel. Interpolating polynomials from their values. *Journal of Symbolic Computation*, Vol. 9, No. 3, pp. 375–403, 1990.